Lab 2

Zara Waheed

4th Feb 2022

Question 6

a)

exp(-6 + 0.05*40 +1*3.5) / (1 + exp(-6 + 0.05*40 +1*3.5))

b)

log(0.5/0.5) = -6 + hours * 0.05 + 3.5 so, log(0.5/0.5) (6 - 3.5)/0.05 # 50 hours

Question 8

We will use the method that has the smaller out-of-sample error rate.

Question 9

a)

x/(1-x) = 0.37 x = 0.37*(1-x) x = 0.37 - 0.37x 1.37x = 0.37 x = 0.37/1.37

b)

0.16/(1-0.16) = 0.19 0.19 are the odds

Question 13

```
weekly <- ISLR2::Weekly
```

a)

```
cor(weekly[, -which(names(weekly) == "Direction")])
```

```
##                Year         Lag1        Lag2        Lag3         Lag4
## Year     1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1    -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2    -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3    -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4    -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5    -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume   0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today   -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                 Lag5      Volume        Today
## Year    -0.030519101  0.84194162 -0.032459894
## Lag1    -0.008183096 -0.06495131 -0.075031842
```

```
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

There doesn't seem to be significant correlation in the data.

b)

```
fit13b <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = weekly, family = "binomial")

summary(fit13b)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Lag2 appears to be statistically insignificant with respect to the p-value.

c)

```
prob13c <- predict(fit13b, type = "response")
pred13c <- rep(times = dim(weekly)[1], x = "Down")
pred13c[prob13c > 0.5] <- "Up"
table(pred13c, weekly[["Direction"]])
```

```
##
## pred13c Down  Up
```

```
##    Down   54  48
##    Up    430 557
```

```
mean(pred13c == weekly[["Direction"]])
```

```
## [1] 0.5610652
```

We can be about 56% sure about the error. The confusion matrix compares the LDA predictions to the true default statuses for the training observations in the Default data set.

d)

```
train <- weekly[["Year"]] <= 2008
test <- weekly[!train, ]
direction <- weekly[["Direction"]][!train]
fit13d <- glm(Direction ~ Lag2,
              data = weekly,
              family = "binomial",
              subset = train)
prob13d <- predict(fit13d,
                    type = "response",
                    newdata = test)

# write a function to look at confusion matrix and accuracy
summ <- function(prob, test_true) {
  pred <- rep(times = length(prob), x = "Down")
  pred[prob > 0.5] <- "Up"

  return( list( cm = table(pred, test_true),
               acc = mean(pred == test_true))
  )
}
summ(prob13d, direction)
```

```
## $cm
##       test_true
## pred   Down Up
##   Down    9  5
##   Up     34 56
##
## $acc
## [1] 0.625
```

e)

```
fit13e <- lda(Direction ~ Lag2,
               data = weekly,
               subset = train)
prob13e <- predict(fit13e,
                    type = "response",
                    newdata = test)
summ(prob13e[["posterior"]][,"Up"], direction)
```

```
## $cm
##       test_true
```

```
## pred   Down Up
##   Down    9  5
##   Up     34 56
##
## $acc
## [1] 0.625
```

f)

```
fit13f <- qda(Direction ~ Lag2,
              data = weekly,
              subset = train)
prob13f <- predict(fit13f, type = "response", newdata = test)
summ(prob13f[["posterior"]][,"Up"], direction)
```

```
## $cm
##     test_true
## pred Down Up
##   Up   43 61
##
## $acc
## [1] 0.5865385
```

g)

```
set.seed(100)
pred13g <- knn( train = as.matrix(weekly[train, "Lag2"]),
                test =  as.matrix(weekly[!train, "Lag2"]),
                cl = weekly[train, "Direction"],
                k = 1 )
(list( cm = table(pred13g, direction),
       acc = mean(pred13g == direction)) )
```

```
## $cm
##        direction
## pred13g Down Up
##    Down   21 29
##    Up     22 32
##
## $acc
## [1] 0.5096154
```

h) Repeat (d) using naive Bayes.

```
fit13h <- naiveBayes(Direction ~ Lag2,
              data = weekly,
              subset = train)
prob13h <- predict(fit13h,
                   type = "raw",
                   newdata = test)
summ(prob13h[,"Up"], direction)
```

```
## $cm
##     test_true
```

```
## pred Down Up
##   Up   43 61
##
## $acc
## [1] 0.5865385
```

i)

LDA seems to provide the best results

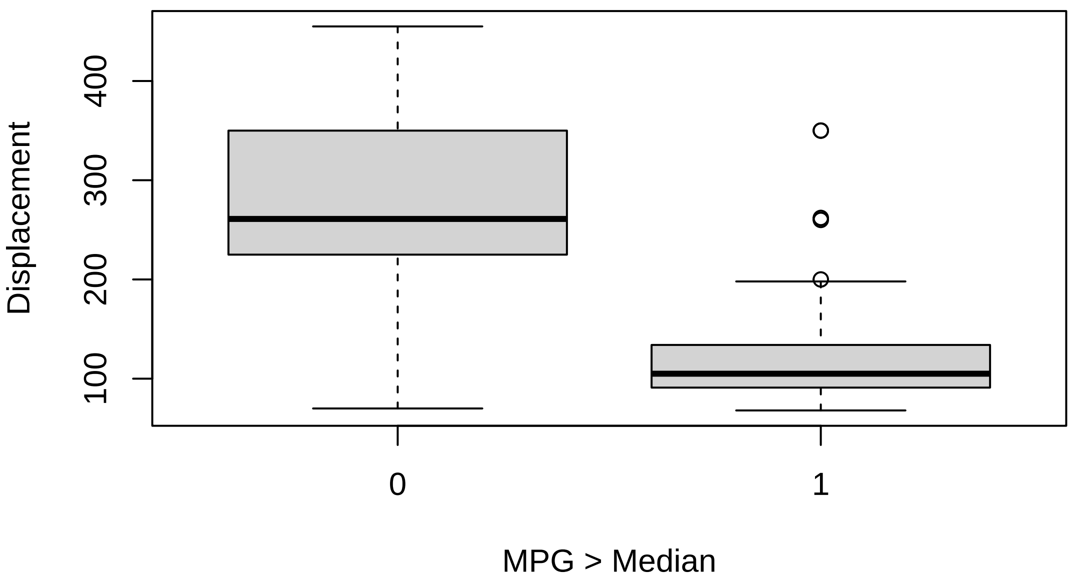
Question 14

a)

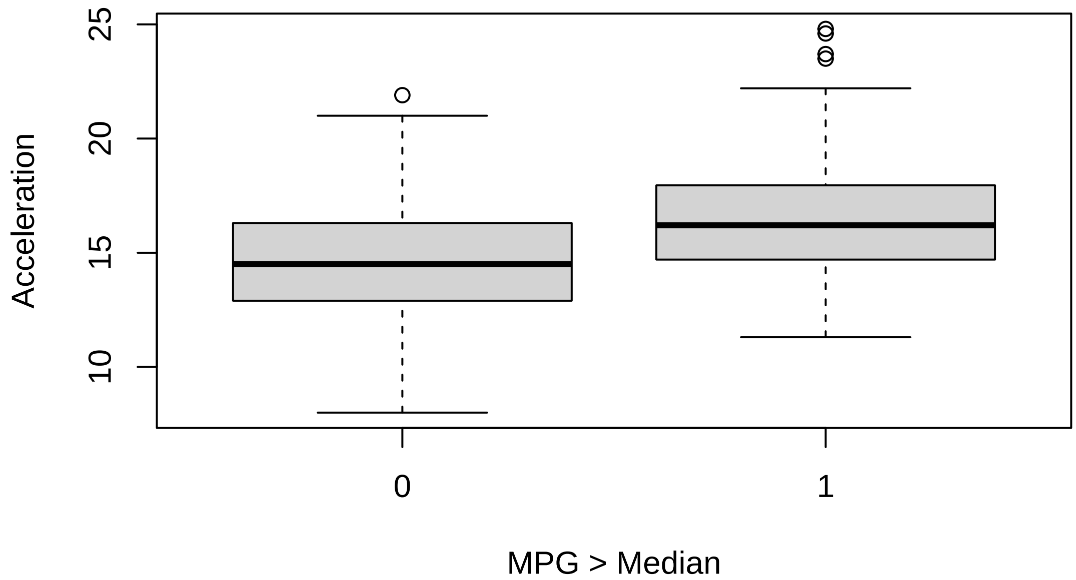
```r
mpg01 = rep(0, nrow(Auto))
mpg01[Auto$mpg >= median(Auto$mpg)] = 1

new.auto = data.frame(Auto, mpg01)
```

b)

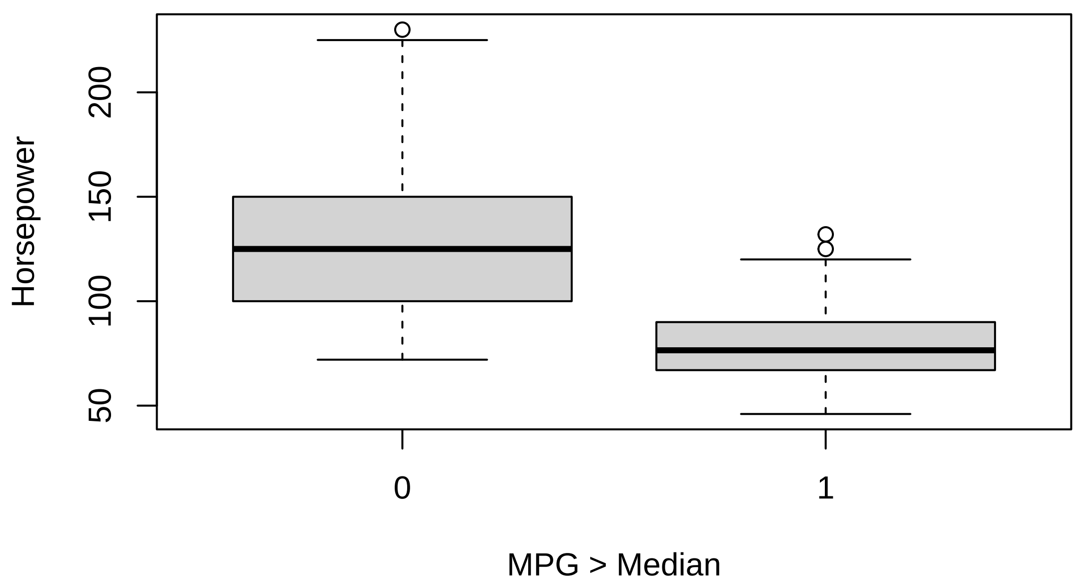
```r
displacement <- boxplot(displacement~mpg01,data=new.auto, xlab="MPG > Median", ylab="Displacement")
```



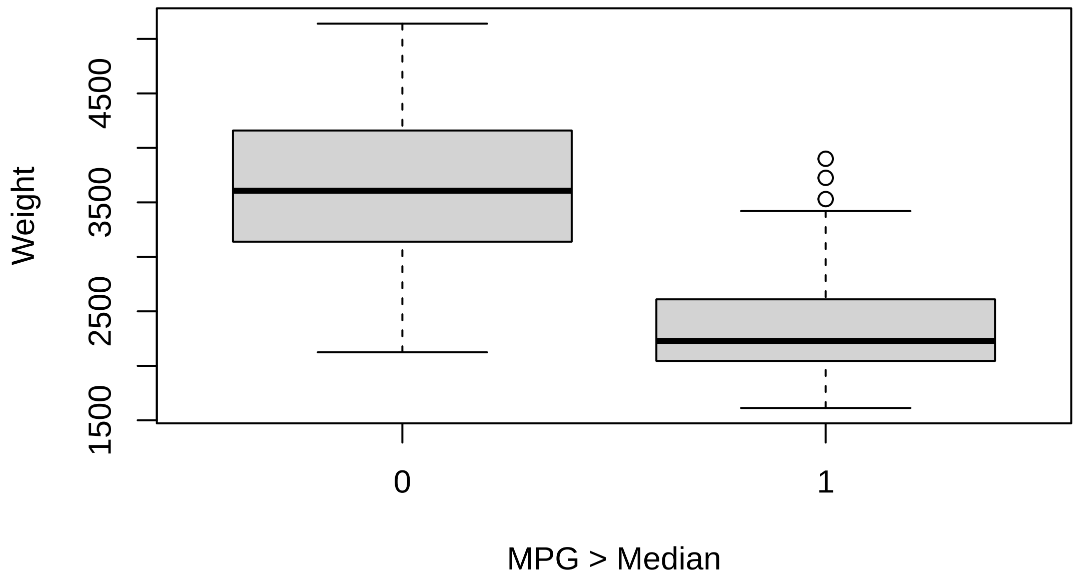

```r
acceleration <- boxplot(acceleration~mpg01,data=new.auto, xlab="MPG > Median", ylab="Acceleration")
```




```
horsepower <- boxplot(horsepower~mpg01,data=new.auto, xlab="MPG > Median", ylab="Horsepower")
```




```
weight <- boxplot(weight~mpg01,data=new.auto, xlab="MPG > Median", ylab="Weight")
```




```
displacement
```

```
## $stats
##      [,1] [,2]
## [1,]   70   68
## [2,]  225   91
## [3,]  261  105
## [4,]  350  134
## [5,]  455  198
##
## $n
## [1] 196 196
##
## $conf
##          [,1]     [,2]
## [1,] 246.8929 100.1471
## [2,] 275.1071 109.8529
##
## $out
## [1] 200 350 260 350 262
##
## $group
## [1] 2 2 2 2 2
##
## $names
## [1] "0" "1"
```

```
acceleration
```

```
## $stats
##      [,1]  [,2]
## [1,]  8.0 11.30
## [2,] 12.9 14.70
## [3,] 14.5 16.20
## [4,] 16.3 17.95
## [5,] 21.0 22.20
##
```

```
## $n
## [1] 196 196
##
## $conf
##          [,1]     [,2]
## [1,] 14.11629 15.83321
## [2,] 14.88371 16.56679
##
## $out
## [1] 21.9 23.5 24.8 23.7 24.6
##
## $group
## [1] 1 2 2 2 2
##
## $names
## [1] "0" "1"
```

```
horsepower
```

```
## $stats
##      [,1]  [,2]
## [1,]   72  46.0
## [2,]  100  67.0
## [3,]  125  76.5
## [4,]  150  90.0
## [5,]  225 120.0
##
## $n
## [1] 196 196
##
## $conf
##          [,1]     [,2]
## [1,] 119.3571 73.90429
## [2,] 130.6429 79.09571
##
## $out
## [1] 230 125 132
##
## $group
## [1] 1 2 2
##
## $names
## [1] "0" "1"
```

```
weight
```

```
## $stats
##        [,1] [,2]
## [1,] 2124.0 1613
## [2,] 3139.5 2045
## [3,] 3607.0 2229
## [4,] 4159.5 2610
## [5,] 5140.0 3420
##
## $n
## [1] 196 196
```

```
##
## $conf
##          [,1]     [,2]
## [1,] 3491.886 2165.236
## [2,] 3722.114 2292.764
##
## $out
## [1] 3530 3900 3725
##
## $group
## [1] 2 2 2
##
## $names
## [1] "0" "1"
```

There seems to be some association with all the variables diplayed above.

c)

```
set.seed(100)
sample = floor(0.8*nrow(new.auto))
train.s = sample(seq_len(nrow(new.auto)), size=sample)

train = new.auto[train.s,]
test = new.auto[-train.s, ]

y.train = train$mpg01
y.test = test$mpg01
```

d)

```
fit14d = lda(mpg01 ~ displacement + horsepower + weight, data=new.auto, subset = train.s)

pred14d = predict(fit14d, test)

# Confusion Matrix
class14d = pred14d$class
table(class14d, y.test)
```

```
##         y.test
## class14d  0  1
##        0 42  2
##        1  4 31
```

```
# error rate
mean(class14d != y.test)
```

```
## [1] 0.07594937
```

e)

```
fit14e = qda(mpg01 ~ displacement + horsepower + weight, data=new.auto, subset = train.s)

pred14e = predict(fit14e, test)
```

```r
# Confusion Matrix
class14e = pred14e$class
table(class14e, y.test)
```

```
##         y.test
## class14e  0  1
##        0 43  2
##        1  3 31
```

```r
# Error rate
mean(class14e != y.test)
```

```
## [1] 0.06329114
```

f)

```r
fit14f = glm(mpg01 ~ displacement + horsepower + weight, data=new.auto, subset = train.s, family=binomi

pred14f = predict(fit14f, test, type="response")

## prediction vector
pred14f = rep(0, nrow(test))
pred14f[pred14f > .5] = 1
table(pred14f, y.test)
```

```
##        y.test
## pred14f  0  1
##       0 46 33
```

```r
# Error rate
mean(pred14f != y.test)
```

```
## [1] 0.4177215
```

h)

```r
# prepare the data
train.x = scale(cbind(train$displacement + train$horsepower + train$weight))
train.y = train$mpg01

test.x = scale(cbind(test$displacement + test$horsepower + test$weight))
```

```r
## KNN for k=1

pred14h.1 = knn(train.x, test.x, train.y, k=1)

# error rate
mean(y.test != pred14h.1)
```

```
## [1] 0.1265823
```

```r
## KNN for k=2

pred14h.2 = knn(train.x, test.x, train.y, k=2)
```

```r
# error rate
mean(y.test != pred14h.2)
```

```
## [1] 0.1518987
```

```r
## KNN for k=3

pred14h.3 = knn(train.x, test.x, train.y, k=3)

# error rate
mean(y.test != pred14h.3)
```

```
## [1] 0.1392405
```

```r
## KNN for k=5

pred14h.5 = knn(train.x, test.x, train.y, k=5)

# error rate
mean(y.test != pred14h.5)
```

```
## [1] 0.07594937
```

```r
## KNN for k=10

pred14h.10 = knn(train.x, test.x, train.y, k=10)

# error rate
mean(y.test != pred14h.10)
```

```
## [1] 0.07594937
```

Question 15

a)

```r
Power <- function(){
  x = 2^3
  print(x)
}
```

b)

```r
Power2 <- function(x,y){
  print(x^y)
}

Power2(3,8)
```

```
## [1] 6561
```

c)

```r
Power2(10,3)
```

```
## [1] 1000
```

```r
Power2(8,17)
```

```
## [1] 2.2518e+15
```

```r
Power2(131,3)
```

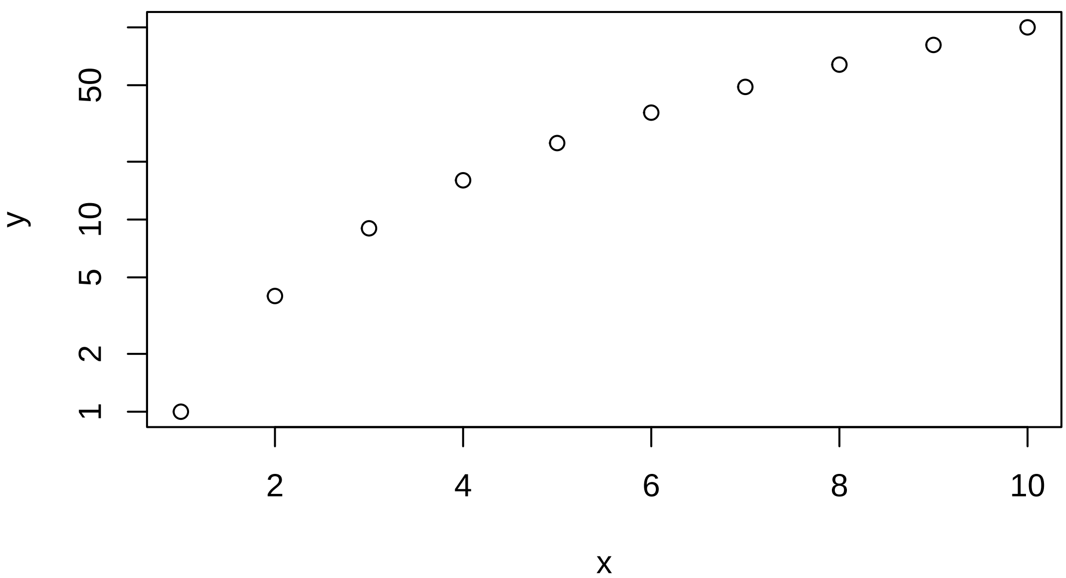
```
## [1] 2248091
```

d)

```r
Power3 <- function(x,y){
  return(x^y)
}
```

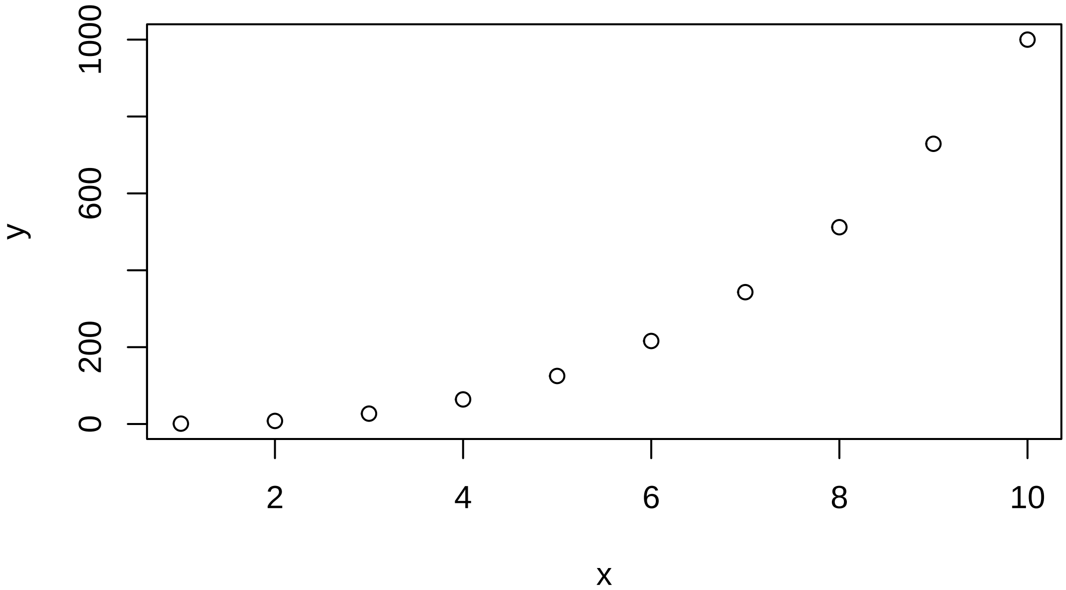
e)

```r
x <- c(1:10)
y <- Power3(x,2)

plot(x,y,log="y")
```



f)

```r
PlotPower <- function(x,a){
  y <- x^a
  plot(x,y)
}

PlotPower(x,3)
```



Question 16

Prepare and explore the data:

```
library(MASS)
attach(Boston)

med.crime = median(crim)

crime = rep(0, length(crim))
crime[crim > med.crime] = 1
crime[1:20]
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
```

```
crime[1:20]
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
```

```
Boston.data = data.frame(Boston, crime)
```

```
# create test and train datasets

train = 1:(dim(Boston)[1]/2)
test = (dim(Boston)[1]/2 + 1):dim(Boston)[1]

Boston.train = Boston.data[train,]
Boston.test = Boston.data[test,]
crime.test = crime[test]
```

LDA

```
set.seed(100)
lda.fit = lda(crime ~indus+nox+age+dis+rad+tax,
              data = Boston, subset = train)
lda.pred = predict(lda.fit, Boston.test)
```

```
lda.class = lda.pred$class
mean(lda.class != crime.test)
```

```
## [1] 0.1067194
```

```
# error = 10.67%
```

QDA

```
set.seed(100)
qda.fit = qda(crime~indus+nox+age+dis+rad+tax,
              data = Boston, subset = train)

qda.class = predict(qda.fit, Boston.test)$class
mean(qda.class != crime.test)
```

```
## [1] 0.6205534
```

```
# error = 62.06%
```

Logistic regression

```
lr.fits = glm(crime~indus+nox+age+dis+rad+tax,
               data = Boston, family = binomial,
               subset = train)

lr.probs = predict(lr.fits, Boston.test, type="response")
lr.pred = rep(0, length(lr.probs))
lr.pred[lr.probs > .5] = 1

mean(lr.pred != crime.test)
```

```
## [1] 0.09090909
```

```
# error = 0.09%
```

KNN

```
# prepare the data
library(class)
train.x = cbind(indus,nox,age,dis,rad,tax)[train, ]
test.x = cbind(indus,nox,age,dis,rad,tax)[test, ]
train.crime = crime[train]
```

```
## KNN for k=1

set.seed(100)
knnpred.1 = knn(train.x, test.x, train.crime, k=1)

# error rate = 62.85%
mean(crime.test != knnpred.1)
```

```
## [1] 0.6284585
```

```r
## KNN for k=5

set.seed(100)
knnpred.5 = knn(train.x, test.x, train.crime, k=5)

# error rate = 12.65%
mean(crime.test != knnpred.5)
```

```
## [1] 0.1264822
```

```r
## KNN for k=10

set.seed(100)
knnpred.10 = knn(train.x, test.x, train.crime, k=10)

# error rate = 11.86%
mean(crime.test != knnpred.10)
```

```
## [1] 0.1185771
```

Logistic regression seems to have the lowest error rate. KNN also looks potentially interesting with low error rates at some values of k.